The Front-end ACNet/LabVIEW Interface

Willem Blokland

Fermilab National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

1. Introduction

This document shows you how to install and customize the portable version of the ACNet/LabVIEW interface. It is recommended that you use this version over the previous Macintosh-only version. The interface enables a LabVIEW application to communicate with a control console program using the ACNet protocol over Ethernet. The tokenring connections are no longer supported in this version. You can still use the older version for this but the current networking plans do not include new tokenring wiring (the Main Injector has no tokenring installed) and it is likely that tokenring will be phased out.

The provided functions enable a console to request single or periodic multiple replies (RETDAT), as well as to send settings to change LabVIEW variables (SETDAT). The interface can also handle Fast Time Plots (FTPMAN) requests at a rate of 15 Hz by redirecting FTPMAN requests to the RETDAT task. The interface can reply up to 200 messages per second depending on computer system and CPU load from other programs.

VIs from the TCPORT interface allow you to read or set any ACNet device in the control system. This interface is described in a separate manual.

LabVIEW is a graphical language complete with analysis library for processing of signals and interfaces to many instrument platforms including VXI. LabVIEW represents a program or subroutine by an icon called Virtual Instrument (VI) which has in- and outputs. The ACNet/LabVIEW Interface providesVIs that setup the communication and access the data. In addition, so-called template VI's demonstrate the use of the Interface VIs and can be customized to fit a particular application as shown in a demo application.

Additional reading on the ACNet/LabVIEW interface can be found in [1] and for applications [2], [3], and [4].

2. Hardware requirements

The ACNet/LabVIEW Interface runs on any LabVIEW supported platform. The computer must have an Ethernet interface.

3. Installing the ACNet/LabVIEW Interface files

3.1. New Installation

To run the ACNet Interface for LabVIEW you must install the following files:

1) The ACNet VI libraries, preferably in the user.lib folder next to the LabVIEW application.

2) The ACNet device table. For the Mac the default is in the in the system folder:Preferences:Your-Project Folder. For other platforms you must define the folder yourself.

3) The file Ac_err.h. This file should be in the Acnet folder next to Your-Project folder. (You don't need this

file to run but it will give you more explanation in the error messages).

3.2. Upgrading from Macintosh only version

Because the VIs that read and write to the Acnet devices have the same calling interface, you can replace the old libraries with the new libraries. When loading your program into LabVIEW make sure LabVIEW finds the new VIs, it will then automatically replace the old VIs with the new VIs. You will have to modify the initialization procedure.

1) Remove all installed files as described to be installed by the old LabVIEW/ACNet document. (You don't have to remove all but you must remove the ACNet VI library).

2) Keep your device tables as they are. The new version reads the same tables.

3) Install the latest version of the file Ac_err.h. (You don't need this file to run but it will give you more explanation in the error messages).

4) In your toplevel VI, modify your initialization setup to conform with the setup described in Setting up your application, and remove the termination setup.

3.3. Getting ACNet to accept you

To get the ACNet system to accept an Ethernet connection and the variables you want to use, take the following, further detailed below, steps:

1) Make up a 6 letter node name,

2) Get an ACNet UDP node number.

3) Get access to the DABBEL account and register LabVIEW variables as ACNet devices .

Make up a node name, e.g. MYNODE, and have it approved and registered by the Controls Group (G. Johnson). Also have ready the IP address of your computer and its Operating System and mention that it will be an Instrumentation node. Now you must get access to DABBEL to enter your node and devices into the General ACNet Database. Again, ask the Controls Group (G. Johnson) to enter you as a user of the DABBEL account.

4. Mapping LabVIEW variables into ACNet devices

You can make DABBEL entries from your VAX account. Now you need to find out how to register your devices with DABBEL. How to do this is described in [5] and [6] obtainable from the Controls Group. ACNet also provides scaling services that you must select in your DABBEL entry, these services and lots more that you will never want to know about are described in [7]. In most cases, you will not have to know how an DABBEL entry looks like, a LabVIEW utility will create these entries for you from your device table, see Appendix A.

An example of a DABBEL entry is as follows:

```
!-------------------------------------
! This file adds a device for the    -
! node INST01, a Mac IIci, to the    -
! data base.                         -
!-------------------------------------
ADD T:BLTVX1 ('Beamline Tuner',INST01)
SSDNHX READNG (0000/0002/0000/0000)
SSDNHX SETTNG (0000/0003/0000/0000)
PRO READNG (2,2,60)
PRO SETTNG (2,2,60)
PDB READNG ('VOLT','BLOK',10,0,2,0,1,0)
PDB SETTNG ('VOLT','BLOK',10,0,2,0,1,0)
! NULL FLTD, X'=X,2 BYTES, INT FORMAT,
! LONG DISPLAY, ADC
```

A device called T:BLTVX1 with the name 'Beamline Tuner' and residing on node INST01 is added to the database. Two properties are registered, one for reading, READNG, and one for writing,

SETTNG. The first property has as SSDN, SubSystem Device Number, 0000/0002/0000/0000. The ACNet functions implemented in LabVIEW will only look at the first two fields, 0000/0002 to identify and ACNet variable. This two word long hexadecimal will be interpreted as a unsigned long integer of value 2 and identifies an ACNet variable in LabVIEW. This is the SSDN number you use to register your variable, see the section on initialization. In LabVIEW, the two properties of the same ACNet device are seen as two different variables as long as they have two different SSDNs. While the ACNet/LabVIEW Interface allows writing and reading to both variables, an ACNet application, such as the parameter page on the console, allows only the setting of the SETTNG property. To be able to use the console to read and write the same variable, give the two properties the same SSDN. The next two words of the SSDN field can be used for offset in case the variable is an array, see appendix A.

The next two lines assign to each of the properties/variables an element size of two bytes and an array size of 2 bytes (one element) as well as a Frequency Time Descriptor of 60 (send an update of the value each 60/60 seconds). Note that the SETTNG property is also read every second. The last two lines declare the units, notation, display type and which ACNet scaling services are to be applied. The DABBEL file can be automatically generated using a general spreadsheet file that is also used to initialize the ACNet/LabVIEW interface, see "Initializing your application" in the next chapter.

5. ACNet/LabVIEW Interface basics

5.1. The main VI

The LabVIEW interface to ACNet is built around the dBase VI, see figure 1.

Figure 1. The dBase VI.

The dBase Vi functions as a database for variable values that must be accessible from both LabVIEW and ACNet. Both sides can read and write values. The dBase VI also logs who accessed at what time. This supports the ACNet requests GTSKID, GTSKNM, GTVERS, GTASKS, GTCNTS, GTSTAT, GTTKST, GTTRIO, and GTPKTS. On the LabVIEW side the VI Ac Statistics can display similar statistics, see section on Ac Statistics.

5.2. Network Access

The ACNet functions communicate directly to LabVIEW. The requests are distributed to the appropriate VIs, one for each task: ACNET, FTPMAN, RETDAT, and SETDAT. Each of these VIs calls the dBase VI for access to the ACNet devices or statistics.

5.3. LabVIEW Access

Because the dBase VI only takes data in the format of the unsigned byte array type, additional VI's, with the naming convention Ac_Read_XXX or Ac_Write_XXX, are available that provide read and write access for specific types of variables.

The XXX specifies the type of variable in question:

1) 'U16': unsigned short integer,

2) 'I16': signed short integer,

3) 'U32': unsigned long integer,

4) 'I32': signed long integer,

5) 'Dbl': double float, 64-bit IEEE format,

6) 'Sgl': single float, 32-bit IEEE format,

7) 'Ext': Extended float, 96-bit MC68882 format,

8) 'Str': String format, defined as array of bytes and must be null terminated.

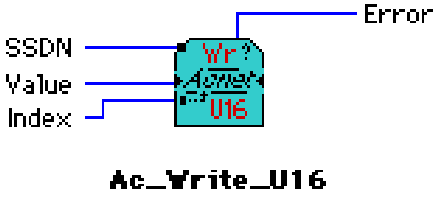


Figure 2. The Ac_Write_U16 VI.

An example of such an access Vi is given in figure 2. Variable access VI are also available for array variables, see figure 3.

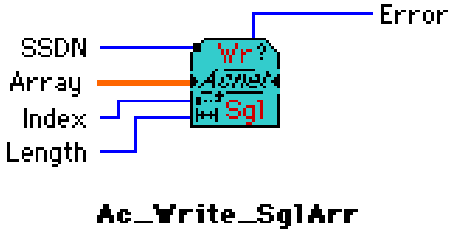


Figure 3. The Ac_Write_SglArr VI.

For easy access to LabVIEW variables from the parameter page of a control console use table 1 to match up a LabVIEW variable type with a specific ACNet scaling service transformation. This table is also used by the VI that automatically generates the DABBEL file.

Table 1. Matching of formats.

LabVIEW	Transformation Index
Sgl	24
U16	20
I16,I32	10

5.4. Error Logging

Errors occur if the requested action cannot be completed. For example, a data request with an SSDN number that is not registered.

1) Interactive. A dialog window is opened, giving the user the option to exit the application or continue. The program will not continue until a button is clicked. See figure 4.

2) Automatic. A window is opened at initialization and any error messages are displayed inside this window, no interaction of the user is required. This window also print progress information during startup. This option is useful for remote application where no user is present and possible errors in writing or reading of data are not fatal to the program. See figure 5.

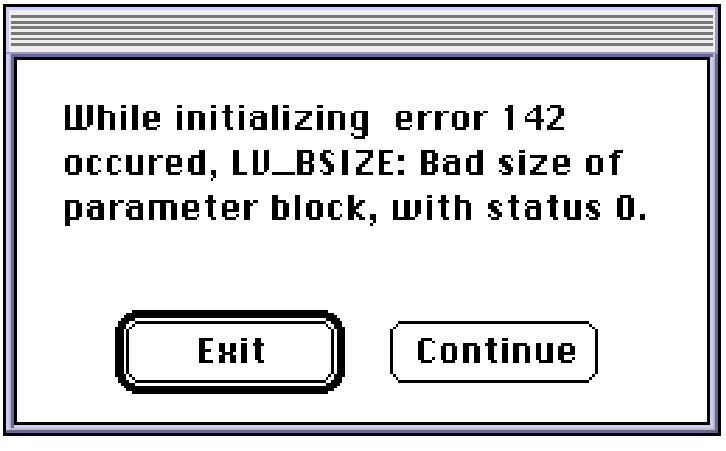


Figure 4. The Dialog error Message.

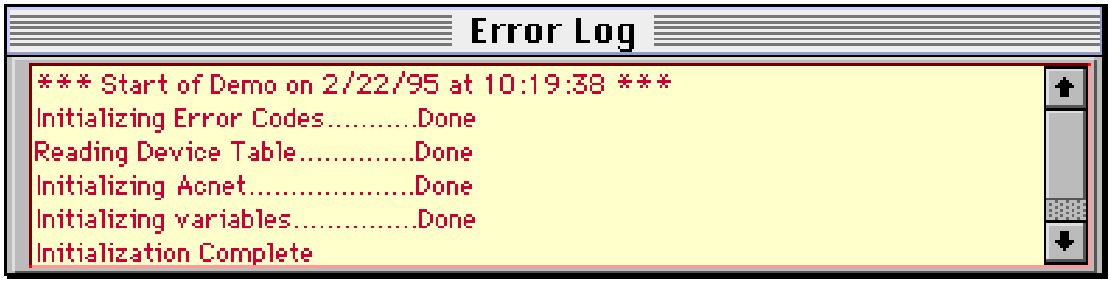


Figure 5. The Error Log

A short explanation is added to the error code if at initialization the VI Ac_InitErrorCodes is executed. This VI reads the Ac_err.h file to associate each error code with a short explanation. The file must be present in the ACNet folder in the Preferences folder. The statistics panel, described later, includes an on-line view of the current status of the Acnet Interface.

6. Setting up your application

A set of template VIs is available that demonstrate the use of the ACNet/LabVIEW interface. These VIs can be customized to fit your application needs. The toplevel VI is the Ac Demo. This VI includes all other template VIs and can be used to encapsulate your application. The demo also includes the TCPORT interface, see TCPORT manual.

6.1. Initializing your application

The ACNet interface must be initialized before and properly terminated after use. The initialization consists of the following actions:

1) Opening a UDP port,

2) Registration of the ACNet accessible variables, see Appendix A,

3) Optional loading of error code file, and

4) Initialization of the ACNet variables and your application (if needed).

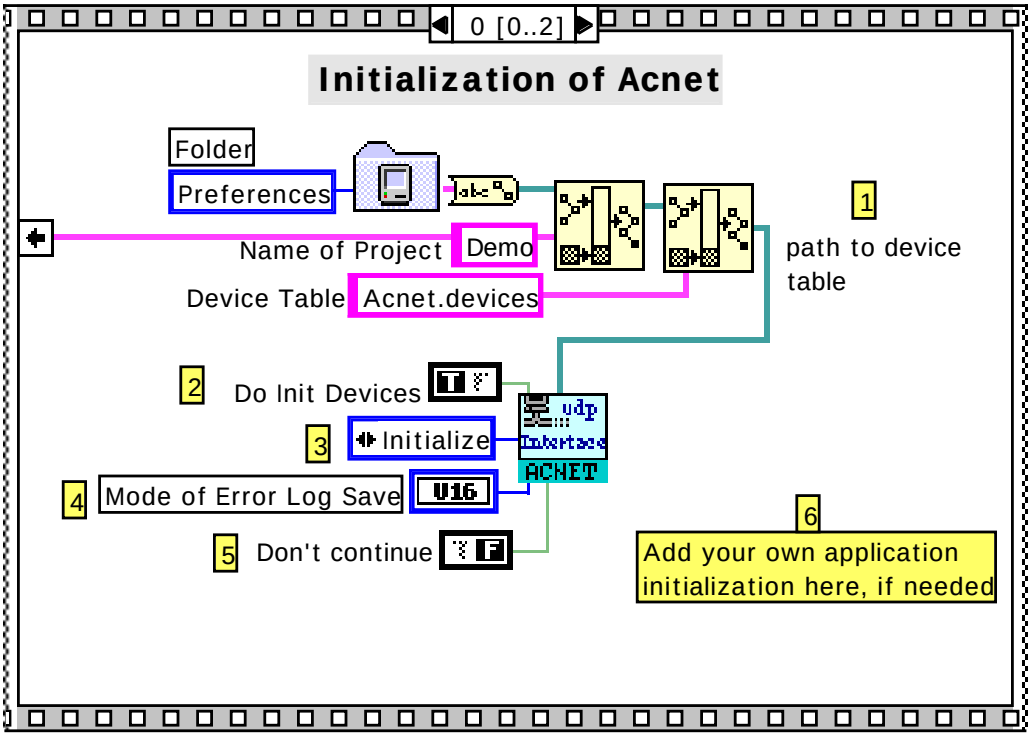


Figure 6. Initialization of ACNet

Figure 6 shows an example of how the initialization is done.

1) Give a path for the device table to use.

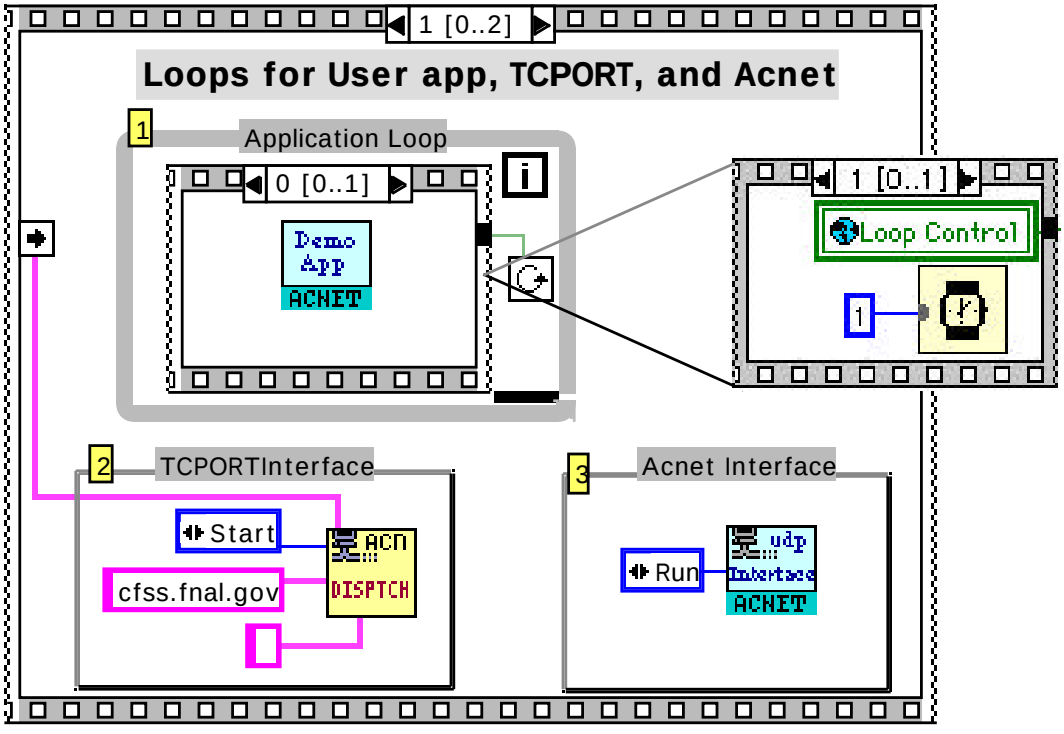
2) Say whether you want the devices to be initialized according to the device table.

3) Give the Initialize command to the Acnet interface.

4) Specify error log display

5) Specify a non-continue. This will initialize Acnet but will not allow communication until you have initialize your application.

6) Add the initialization, if you have one, of your application.



6.1.1.

Figure 7. Running your application.

6.2. Running your application

Figure 7 shows the second frame of the Acnet Demo VI containing a while loops (1) for your application VI, the TCPORT Interface (2) which has its own internal while loop, and the Acnet interface VI (3) which also has its own while loop. All loops should terminate when Loop Control global is set to false. Your application should have a stop button which will set this global to false. LabVIEW executes all loops, or any other added, in parallel so that your program, the Acnet interface and if included the TCPORT interface all run at the same time. Remember to add the Wait ms VI to each loop to prevent it from hogging all the CPU time. (Note that different platforms have different threading capabilities that can assist you in setting up the time-slicing. See the LabVIEW users manual for details)

6.3. The Interface Statistics

The display of the statistics panel is shown in figure 8. The statistics page allows you to see information about the ACNet interface:

1) Time: The current time in milliseconds

2) Timers: List of active timers, each entry has the next wakeup time, period and number of elements using the timer.

3) Active Requests: Lists the current periodic requests: Acnet Header, Drb Header, drb blocks, and Network information about requester and used port.

4) Device Statistics: Table of the Acnet devices that are currently registered. The columns have the device name, SSDN, Variable Type, Size (bytes), Reads Local, Sets Local, Reads over Network (Acnet), Sets

over Network, and time stamp of last write.

5) Task Statistics: List of each registered task with the number of: Info (not used), Multiple reply request with Periods, Single reply request with Periods, Multiple reply request on Event, Single reply request on Event, Cancellation requests, Errors, USM received, Requests received, Replies received, USM transmitted, Requests transmitted, Replies transmitted.

6) Boot: Dat and time of booting the interface.

7) Version: Software version.

The requests are organized by FTD value (each FTD value gets its own timer).

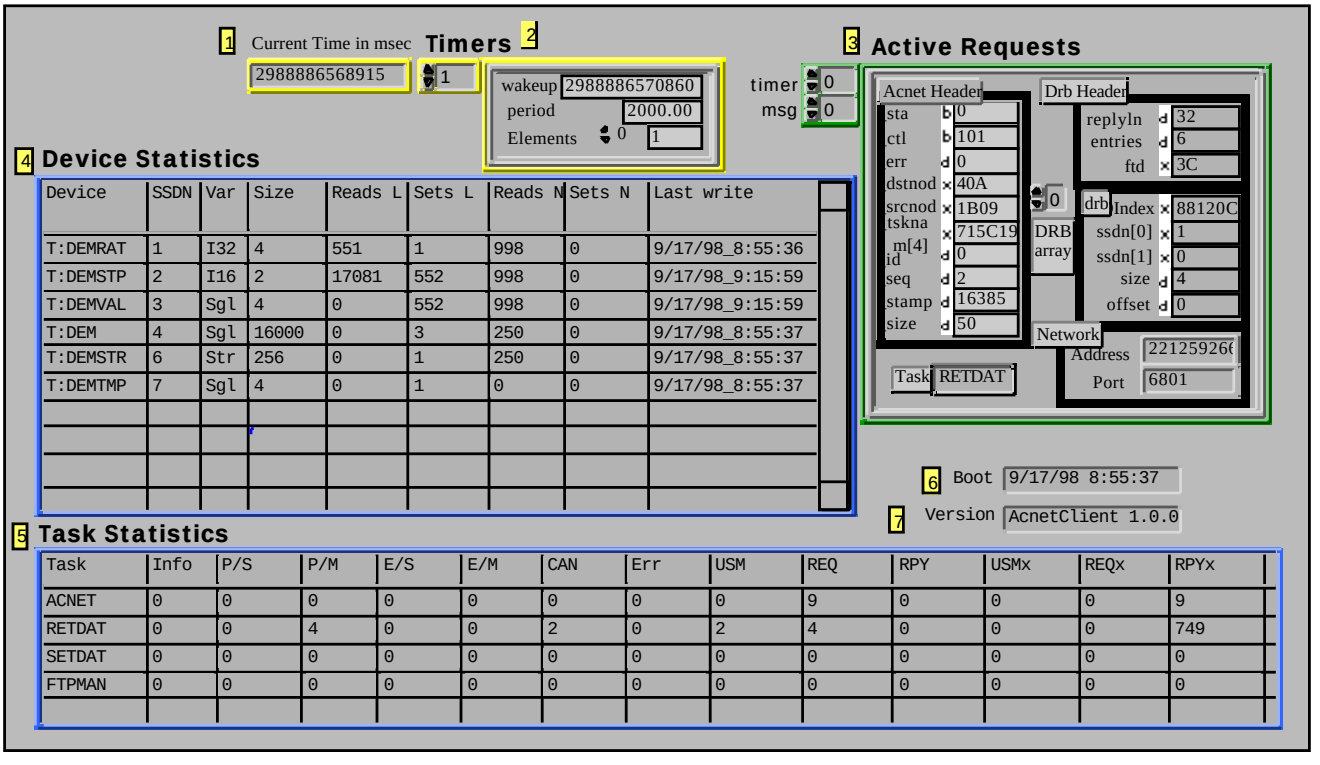


Figure 9. The Statistics panel.

6.4. Terminating your application

The ACNet functions are terminated by setting the Loop Control global to false. In the third frame of the Acnet Demo VI is space to place your VI for exiting instruments or cleaning up any data structures.

7. References

[1] W. Blokland, "An Interface from LabVIEW to the Accelerator Controls Network," Conf. Proc. of the Accel. Instr. Third Annual Workshop, LBL, Berkeley, 1992.

[2] W. Blokland, "A VXI/LabVIEW-based Beamline Tuner." PAC 93.

[3] A. A. Hahn and P. Hurh, "Results from a Imaging Beam Monitor in the Tevatron using Synchrotron Light", presented at the XVth Int'l Conf. on

High Energy Accel., Hamburg, Germany, 1991.

[4] W. Blokland, 'Integrating the commercial software package LabVIEW with Fermilab's Accelerator Control NETwork', ICALEPS 95, pp. 226-234, Oct. 29 - Nov. 3, Chicago, USA.

[5] G. C Johnson, "Using the DABBEL Turnkey Account." Controls Software Release No. 120.0. Fermilab, Batavia IL, November 7 1983.

[6] S. Sommers, "DABBEL (DataBase Batch Editing Language)." Controls Software Release No. 119.1. Fermilab, Batavia IL, April 10 1984.

[7] K. Cahill et. al, "ACNet Data Acquisition, Scaling and Data-base Services." ACNET Design Note 22.28, Fermilab, Batavia IL, December 6 1985.

8. APPENDIX A: Automatic Registration of ACNet devices in LabVIEW

8.1. The Device Table

Information about the ACNet devices should be presented in a table format as made by spreadsheet programs such as Excel or word processors such as Word. The final version of the table should be saved as a text file in which the elements of a row are separated by tabs and the row is ended by a carriage return. The file should be located in the Preferences:YourProject: folder of the System Folder. An example is given below in Table A1.

3/27/97	Example	V 2.0									W. Blokland
Device name	SSDN#	Read/ Write	Var type	Bytes/ elem	Elem/ device	Initial value	Update rate	Prim Unit	Comm Unit	Long/ Short	Notation	Description
T:SCALAR	1	RW	SGL	4	1	269.1	60	mVlt	DegC	short	dec	Scalar device
T:ARRAY	2	RW	SGL	4	100	1	120	mm	mm	short	dec	Array device
T:DEM	100@	RW	I32	4	4000	0	60	Volt	Volt	short	dec	Full array for LabVIEW
T:ALIAS1	100[0]	RW	I32	4	2000	0	60	Volt	Volt	short	dec	First alias for Acnet
T:ALIAS2	100[2000]	R	I32	4	2000	0	60	Volt	Volt	short	dec	Second alias for Acnet
T:ARR	200	R	I16	2	10	1	60	amps	amps	short	dec	Full array for LabVIEW
T:ARR%02d	200[1-9]	R	I16	2	1	0	60	amps	amps	short	dec	Alias[%1d] for Acnet

Table A1. An example of ACNet device entries.

The first two rows contain information other than device entries. The first row should contain the date that the table was created, the project name, and the version number. The next row contains the headers to identify each column. All following rows must contain information about the devices. The elements must be as follows:

Device name : The ACNet device name with optional format string for use with indexing.

SSDN# : The SubSystem Device Number with optional index number, [number], or index range, [number1-number2] number2 > number1. Both the SSDN and the index are numbers from 0 to 32768 (2^{15}). The SSDN identifies the LabVIEW variable, the index the offset (in elements) in the array if the variable is an array device. An '@' can be postfixed to SSDN entry without additional indices to indicated to not generate a DABBEL entry for this device entry.

Read/Write : The ACNet device property. Values are: R for ACNet read access, W for write access and RW for both accesses.

Var type : The variable type as defined in the "Accessing ACNet variables" section.

Bytes/elem : The number of bytes per element.

Elem/device : The number of elements per device.

Initial value : The initial value of the LabVIEW variable can be a number or a string. In the case that the variable is an array, the whole array is initialized with this value. Entries that are DABBEL aliases are not used to initialize devices in the labVIEW program.

Update rate : The Update rate (the Frequency Time Descriptor). The rate in seconds is the Update rate divided by 60.

Prim unit : Primary unit used for parameter page display. Four letters maximum, no commas or quotes.

Com unit : Common unit used for parameter page display. Four letters maximum, no commas or quotes.

Long/Short : Notation length of value on parameter page: LONG or SHORT.

Notation : Scientific or decimal notation of value on parameter page: DEC or SCI.

Description : Description of value for parameter page with optional format string for use with indexing. Maximum of 24 characters after formatting.

Device names with indexing (called an alias device), e.g. T:DEM1, will not lead to the creation of a new LabVIEW variable. These devices are solely for console display purposes to define a scalar device entry from an array device or to split a very large array into two or more smaller entries to bypass Acnet's size limititations. Since LabVIEW has no trouble accessing large array, it will ignore the DABBEL alias device entries and thus ignore the inital values of these entries. When splitting large arrays in pieces, you must first define the device for the LabVIEW program, if you don't want this to generate a DABBEL entry then postfix it with a '@', see the example table.

8.2. Generation of the DABBEL File

The DABBEL file is generated by the LabVIEW VI Ac MakeDabbel. This VI converts the device table to a DABBEL file. You must also declare the Node name, e.g. INST08, on which the ACNet devices will be located. Additionally, you must indicate whether you are adding or modifying ACNet devices. Running the VI will produce the DABBEL text file, This file must then be copied to the DABBEL host computer and run through the DABBEL program to add the ACNet devices into the control systems database.

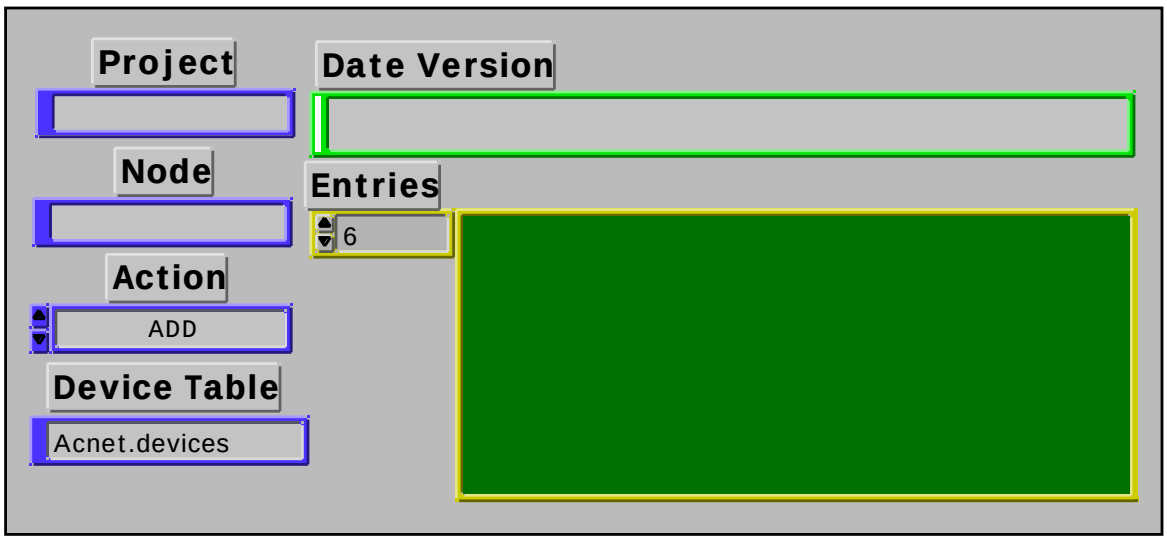


Figure A1. The Ac MakeDabbel VI.

If the declared device has an index or a range of indices then this single table entry will generate a DABBEL device entry for each number. For example, an SSDN of 200[1-9] will generate 9 devices with SSDN 200 and an index from 1 to 9 inclusive. In the same row, one can refer to the index using a for-

mat string comparable to a printf statement in the C language. For example, the device name T:ARR%O2d will be formatted as a three digit, left zero-padded number to T:ARR01, T:ARR02, ..., T:ARR09. Any format statement will be expanded with the value of the index. This means that the descriptive text is also formatted to Alias1],.....,Alias[9]. See the next page for the generated file. Note that in the SSDNHX line the SSDN is in the second field while the fourth field contains the hexadecimal offset in bytes into the array. In the case, of T:ARR04 this becomes: index∗element_size equals 4∗2 = 0x0008.

```
! FILE : Blok:Desktop Folder:Demo.dab
! FROM : 3/27/97 Example V 2.0 W. Blokland
! Created by LV converter 3.1

ADD T:SCALAR ('Scalar device', MYNODE)
SSDNHX READNG (0000/0001/0000/0000)
SSDNHX SETTNG (0000/0001/0000/0000)
PRO READNG (4,4,60)
PRO SETTNG (4,4,60)
PDB READNG ('mVlt','DegC',24,0,4,0,0,0)
PDB SETTNG ('mVlt','DegC',24,0,4,0,0,0)

ADD T:ARRAY ('Array device', MYNODE)
SSDNHX READNG (0000/0002/0000/0000)
SSDNHX SETTNG (0000/0002/0000/0000)
PRO READNG (4,400,120)
PRO SETTNG (4,400,120)
PDB READNG ('mm','mm',24,0,4,0,0,0)
PDB SETTNG ('mm','mm',24,0,4,0,0,0)

ADD T:ALIAS1 ('First alias for Acnet', MYNODE)
SSDNHX READNG (0000/0064/0000/0000)
SSDNHX SETTNG (0000/0064/0000/0000)
PRO READNG (4,8000,60)
PRO SETTNG (4,8000,60)
PDB READNG ('Volt','Volt',10,0,4,0,0,0)
PDB SETTNG ('Volt','Volt',10,0,4,0,0,0)

ADD T:ALIAS2 ('Second alias for Acnet',
MYNODE)
SSDNHX READNG (0000/0064/0000/1F40)
PRO READNG (4,8000,60)
PDB READNG ('Volt','Volt',10,0,4,0,0,0)

ADD T:ARR ('Full array for LabVIEW',
MYNODE)
SSDNHX READNG (0000/00C8/0000/0000)
PRO READNG (2,20,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR01 ('Alias[1] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0002)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR02 ('Alias[2] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0004)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR03 ('Alias[3] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0006)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR04 ('Alias[4] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0008)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR05 ('Alias[5] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/000A)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR06 ('Alias[6] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/000C)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR07 ('Alias[7] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/000E)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR08 ('Alias[8] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0010)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)

ADD T:ARR09 ('Alias[9] for Acnet', MYNODE)
SSDNHX READNG (0000/00C8/0000/0012)
PRO READNG (2,2,60)
PDB READNG ('amps','amps',10,0,2,0,0,0)
```